



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ  
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики  
Кафедра програмного забезпечення комп'ютерних систем

**Лабораторна робота №2**  
з дисципліни «Бази даних»

тема «Створення додатку бази даних, орієнтованого на взаємодію з СУБД  
PostgreSQL»

Виконав(ла)  
студент(ка) II курсу  
групи КП-03

Сітайло Анна Сергіївна  
(*прізвище, ім'я, по батькові*)

Варіант №17

Перевірів  
“ \_\_\_\_ ” “ \_\_\_\_\_ ” 20\_\_ р.  
викладач

Радченко Костянтин  
Олександрович  
(*прізвище, ім'я, по батькові*)

## Мета

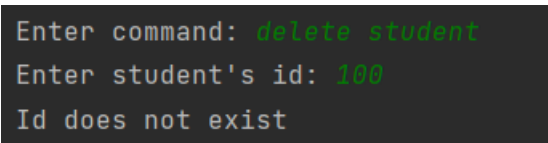
Здобуття вмінь програмування прикладних додатків баз даних PostgreSQL.

## Завдання

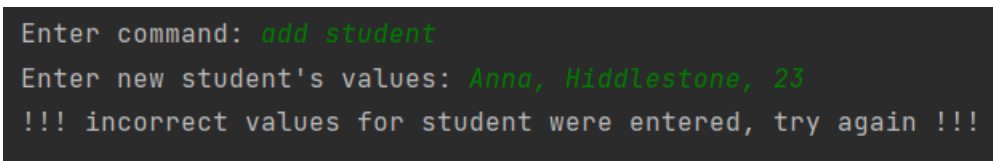
1. Реалізувати функції внесення, редагування та вилучення даних у таблицях бази даних, створених у лабораторній роботі No1, засобами консольного інтерфейсу.
2. Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі.
3. Забезпечити реалізацію пошуку за декількома атрибутами з двох та більше сутностей одночасно: для числових атрибутів – у рамках діапазону, для рядкових – як шаблон функції LIKE оператора SELECT SQL, для логічного типу – значення True/False, для дат – у рамках діапазону дат.
4. Програмний код виконати згідно шаблону MVC (модель-подання-контролер).

## Результати

1. Вимоги до 1-го пункту деталізованого завдання

a. 

```
Enter command: delete student
Enter student's id: 100
Id does not exist
```

b. 

```
Enter command: add student
Enter new student's values: Anna, Hiddlestone, 23
!!! incorrect values for student were entered, try again !!!
```

2. Вимоги до 2-го пункту деталізованого завдання

Query Editor

Query History

1

SELECT

\*

FROM

scd

ule

Data Output

Explain

Messages

Notifications

	id [PK] integer	day text	time text	subject_teacher_id integer	student_id integer
1	1	monday	8:30	1	1
2	2	monday	9:25	5	1
3	3	monday	10:20	3	2
4	4	tuesday	8:30	2	3
5	12	saturday	7:3	2	2
6	13	thursday	1:1	1	1
7	14	friday	9:12	1	1
8	15	wedness...	5:13	2	2
9	16	saturday	14:14	1	2
10	17	thursday	5:11	2	1
11	18	tuesday	9:11	1	2
12	19	saturday	7:14	2	2
13	20	saturday	14:9	2	2
14	21	thursday	11:10	1	1
15	22	friday	7:13	2	1
16	23	saturday	6:5	2	2
17	24	saturday	10:5	1	2
18	25	tuesday	6:10	2	1
19	26	wednesday	12:12	1	2

Query Editor

Query History

1

SELECT

\*

FROM

subjects

Data Output

Explain

Messages

Notifications

	id [PK] integer	name text	classes_per_semester integer
1	1	math	36
2	2	english	18
3	3	chemistry	27
4	4	FE	9
5	5	OH	6
6	6	AD	7
7	7	KP	9
8	8	HJ	11
9	9	BF	6
10	10	MG	7
11	11	OB	11
12	12	DN	8
13	13	YU	7
14	14	JH	14
15	15	IL	14
16	16	KX	9
17	17	OX	6
18	18	RF	9
19	19	PG	6

Query Editor

Query History

1

SELECT

\*

FROM

students

Data Output

Explain

Messages

Notifications

	id [PK] integer	name text	age integer	grade text
49	52	AN	14	7
50	53	RI	10	6
51	54	YI	12	5
52	55	WQ	13	6
53	56	II	20	0
54	57	YX	8	7
55	58	YU	13	21
56	59	YN	29	34
57	60	QG	19	36
58	61	RB	14	46
59	62	JF	16	15
60	63	HE	17	12
61	64	HH	18	3
62	65	HS	11	5
63	66	IO	19	4
64	67	LL	19	5
65	68	MW	8	8
66	69	KU	20	3
67	70	PS	6	6
68	71	WM	14	4

Query Editor

Query History

1

SELECT

\*

FROM

teachers

Data Output

Explain

Messages

Notifications

	id [PK] integer	name text	age integer	work_experience integer
1	1	Chris Letherwood	28	3
2	2	Abigail Swan	45	15
3	3	Landon Smith	36	11
4	4	BC	15	17
5	5	XQ	6	10
6	6	TC	10	11
7	7	JL	10	11
8	8	OO	8	19
9	9	RS	15	14
10	10	QY	9	17
11	11	SG	14	16
12	12	NM	8	15
13	13	IJ	8	18

Query Editor

Query History




1SELECT \* FROM subjects\_teachers

Data Output

Explain

Messages

Notifications

	 id [PK] integer	 subject_id integer	 teacher_id integer
1	1	1	2
2	2	3	2
3	3	1	1
4	4	2	3
5	5	2	1
6	46	10	11
7	47	11	9
8	48	9	7
9	49	7	8
10	50	10	9
11	51	6	12
12	52	8	3
13	53	7	12
14	54	6	3
15	55	6	11
16	56	9	3
17	57	15	10
18	58	9	12
19	59	11	10

Query Editor

Query History





1 SELECT \* FROM marks

Data Output

Explain

Messages

Notifications

	 id [PK] integer	 student_id integer	 subject_teacher_id integer	 mark integer
1	1	1	2	10
2	2	2	1	12
3	3	3	4	11
4	4	2	2	14
5	5	1	2	5
6	6	1	2	13
7	7	1	1	6
8	8	1	1	14
9	9	1	2	5
10	10	2	1	6
11	11	2	2	10
12	12	2	1	10
13	13	1	2	13
14	14	1	1	11
15	15	2	2	8
16	16	1	2	6
17	17	2	1	11
18	18	2	2	11
19	19	1	2	10

### 3. Вимоги до 3-го пункту деталізованого завдання

```

Enter command: search subjects_teachers records by name, classes, age
Enter classes per semester range: 1, 40
Enter subject name: math
Enter teacher age range: 20, 40
_____search started_____
('math', 36, 'Chris Letherwood', 28)
('math', 36, 'Landon Smith', 36)
_____search ended_____

```

```
Enter command: search schedule records by id, name, day
Enter id range: 2, 30
Enter teacher's name: Abigail
Enter day: monday
_____search started_____
(27, 'monday', '14:9', 'math', 'Abigail Swan', 'Tom White')
_____search ended_____
```

```
Enter command: search marks records by name, mark, grade
Enter mark range: 1, 6
Enter subject name: math
Enter grade: 7
_____search started_____
(6, 'Tom White', '7', 'math', 'Abigail Swan')
_____search ended_____
```

4. Вимоги до 4-го пункту деталізованого завдання

Main.py:

5 lines (4 sloc) | 112 Bytes

```
1  import controller
2
3  if __name__ == "__main__":
4      command = input("Enter command: ")
5      controller.run(command)
```

## Controller.py:

289 lines (282 sloc) | 12 KB

```
1  import psycopg2
2  from psycopg2 import OperationalError
3  from view import View
4  from model import Student, Teacher, Subject, Subject_Teacher, Schedule, Mark, Search
5
6
7
8  def create_connection():
9      connection = None
10     try:
11         connection = psycopg2.connect(
12             database='postgres',
13             user='postgres',
14             password=1234567890,
15             host='localhost',
16             port=5432,
17         )
18     except OperationalError as e:
19         print(f"The error '{e}' occurred")
20     return connection
21
22 def run(command):
23     connection = create_connection()
24
25     student = Student(connection)
26     teacher = Teacher(connection)
27     subject = Subject(connection)
28     subject_teacher = Subject_Teacher(connection)
29     schedule = Schedule(connection)
30     mark = Mark(connection)
31     search = Search(connection)
32
33     view = View()
34
35     if command == "add student":
36         try:
37             command_line = view.ask_for_values_to_add("student")
38             v1 = command_line.split(", ")[0]
39             v2 = int(command_line.split(", ")[1])
40             v3 = int(command_line.split(", ")[2])
41             student.add_student(command_line)
```

```

37         command_line = view.ask_for_values_to_add("student")
38         v1 = command_line.split(", ")[0]
39         v2 = int(command_line.split(", ")[1])
40         v3 = int(command_line.split(", ")[2])
41         student.add_student(command_line)
42         view.added_message("student")
43     except:
44         view.incorrect_input_message("student")
45 elif command == "update student":
46     try:
47         command_line = view.ask_for_values_to_update("student")
48         v1 = command_line.split(", ")[0]
49         v2 = int(command_line.split(", ")[1])
50         v3 = int(command_line.split(", ")[2])
51         student.update_student(command_line)
52         view.updated_message("student")
53     except:
54         view.incorrect_input_message("student")
55 elif command == "delete student":
56     try:
57         command_line = int(view.ask_for_values_to_delete("student"))
58         student.delete_student(command_line)
59         view.deleted_message("student")
60     except:
61         view.incorrect_input_message("student")
62 elif command == "get random students":
63     try:
64         command_line = int(view.ask_for_values_to_generate("student"))
65         student.generate_random_students(command_line)
66         view.generated_message("student")
67     except:
68         view.incorrect_input_message("student")
69 elif command == "add teacher":
70     try:
71         command_line = view.ask_for_values_to_add("teacher")
72         v1 = command_line.split(", ")[0]
73         v2 = int(command_line.split(", ")[1])
74         v3 = int(command_line.split(", ")[2])
75         teacher.add_teacher(command_line)
76         view.added_message("teacher")
77     except:
78         view.incorrect_input_message("teacher")
79 elif command == "update teacher":
80     try:
81         command_line = view.ask_for_values_to_update("teacher")

```

```

81         command_line = view.ask_for_values_to_update("teacher")
82         v1 = command_line.split(", ")[0]
83         v2 = int(command_line.split(", ")[1])
84         v3 = int(command_line.split(", ")[2])
85         teacher.update_teacher(command_line)
86         view.updated_message("teacher")
87     except:
88         view.incorrect_input_message("teacher")
89 elif command == "delete teacher":
90     try:
91         command_line = int(view.ask_for_values_to_delete("teacher"))
92         teacher.delete_teacher(command_line)
93         view.deleted_message("teacher")
94     except:
95         view.incorrect_input_message("teacher")
96 elif command == "get random teachers":
97     try:
98         command_line = int(view.ask_for_values_to_generate("teacher"))
99         teacher.generate_random_teachers(command_line)
100        view.generated_message("teacher")
101    except:
102        view.incorrect_input_message("teacher")
103 elif command == "add subject":
104     try:
105         command_line = view.ask_for_values_to_add("subject")
106         v1 = command_line.split(", ")[0]
107         v2 = int(command_line.split(", ")[1])
108         subject.add_subject(command_line)
109         view.added_message("subject")
110     except:
111         view.incorrect_input_message("subject")
112 elif command == "update subject":
113     try:
114         command_line = view.ask_for_values_to_update("subject")
115         v1 = command_line.split(", ")[0]
116         v2 = int(command_line.split(", ")[1])
117         subject.update_subject(command_line)
118         view.updated_message("subject")
119     except:
120         view.incorrect_input_message("subject")
121 elif command == "delete subject":
122     try:
123         command_line = view.ask_for_values_to_delete("subject")
124         v = int(command_line)
125         subject.delete_subject(command_line)

```



```

125         subject.delete_subject(command_line)
126         view.deleted_message("subject")
127     except:
128         view.incorrect_input_message("subject")
129 elif command == "get random subjects":
130     try:
131         command_line = view.ask_for_values_to_generate("subject")
132         v = int(command_line)
133         subject.generate_random_subjects(command_line)
134         view.generated_message("subject")
135     except:
136         view.incorrect_input_message("subject")
137 elif command == "add subjects_teachers record":
138     try:
139         command_line = view.ask_for_values_to_add("subjects_teachers record")
140         v1 = int(command_line.split(", ")[0])
141         v2 = int(command_line.split(", ")[1])
142         subject_teacher.add_subjects_teachers_record(command_line)
143         view.added_message("subjects_teachers record")
144     except:
145         view.incorrect_input_message("subjects_teachers record")
146 elif command == "update subjects_teachers record":
147     try:
148         command_line = view.ask_for_values_to_update("subjects_teachers record")
149         v1 = int(command_line.split(", ")[0])
150         v2 = int(command_line.split(", ")[1])
151         subject_teacher.update_subjects_teachers_record(command_line)
152         view.updated_message("subjects_teachers record")
153     except:
154         view.incorrect_input_message("subjects_teachers record")
155 elif command == "delete subjects_teachers record":
156     try:
157         command_line = view.ask_for_values_to_delete("subjects_teachers record")
158         v = int(command_line)
159         subject_teacher.delete_subjects_teachers_record(command_line)
160         view.deleted_message("subjects_teachers record")
161     except:
162         view.incorrect_input_message("subjects_teachers record")
163 elif command == "get random subjects_teachers records":
164     try:
165         command_line = view.ask_for_values_to_generate("subjects_teachers record")
166         v = int(command_line)
167         subject_teacher.generate_random_subjects_teachers_records(command_line)
168         view.generated_message("subjects_teachers record")
169     except:

```

```

169         except:
170             view.incorrect_input_message("subjects_teachers record")
171     elif command == "add schedule record":
172         try:
173             command_line = view.ask_for_values_to_add("schedule")
174             v1 = command_line.split(", ")[0]
175             v2 = command_line.split(", ")[1]
176             v3 = int(command_line.split(", ")[2])
177             v4 = int(command_line.split(", ")[3])
178             schedule.add_schedule_record(command_line)
179             view.added_message("schedule")
180         except:
181             view.incorrect_input_message("schedule")
182     elif command == "update schedule record":
183         try:
184             command_line = view.ask_for_values_to_update("schedule")
185             v1 = command_line.split(", ")[0]
186             v2 = command_line.split(", ")[1]
187             v3 = int(command_line.split(", ")[2])
188             v4 = int(command_line.split(", ")[3])
189             schedule.update_schedule_record(command_line)
190             view.updated_message("schedule")
191         except:
192             view.incorrect_input_message("schedule")
193     elif command == "delete schedule record":
194         try:
195             command_line = view.ask_for_values_to_delete("schedule")
196             v = int(command_line)
197             schedule.delete_schedule_record(command_line)
198             view.deleted_message("schedule")
199         except:
200             view.incorrect_input_message("schedule")
201     elif command == "get random schedule records":
202         try:
203             command_line = view.ask_for_values_to_generate("schedule")
204             v = int(command_line)
205             schedule.generate_random_schedule_records(command_line)
206             view.generated_message("schedule")
207         except:
208             view.incorrect_input_message("schedule")
209     elif command == "add mark record":
210         try:
211             command_line = view.ask_for_values_to_add("mark record")
212             v1 = int(command_line.split(", ")[0])
213             v2 = int(command_line.split(", ")[1])

```

```

213         v2 = int(command_line.split(", ")[1])
214         v3 = int(command_line.split(", ")[2])
215         mark.add_mark(command_line)
216         view.added_message("mark record")
217     except:
218         view.incorrect_input_message("mark record")
219 elif command == "update mark record":
220     try:
221         command_line = view.ask_for_values_to_update("mark record")
222         v1 = int(command_line.split(", ")[0])
223         v2 = int(command_line.split(", ")[1])
224         v3 = int(command_line.split(", ")[2])
225         mark.update_mark(command_line)
226         view.updated_message("mark record")
227     except:
228         view.incorrect_input_message("mark record")
229 elif command == "delete mark record":
230     try:
231         command_line = view.ask_for_values_to_delete("mark record")
232         v = int(command_line)
233         mark.delete_mark(command_line)
234         view.deleted_message("mark record")
235     except:
236         view.incorrect_input_message("mark record")
237 elif command == "get random mark records":
238     try:
239         command_line = view.ask_for_values_to_generate("mark record")
240         v = int(command_line)
241         mark.generate_random_marks(command_line)
242         view.generated_message("mark record")
243     except:
244         view.incorrect_input_message("mark record")
245 elif command == "search subjects_teachers records by name, classes, age":
246     try:
247         classes_range = view.ask_for_values_to_search("classes per semester range").split(", ")
248         subject_name = view.ask_for_values_to_search("subject name")
249         age_range = view.ask_for_values_to_search("teacher age range").split(", ")
250         v1 = int(classes_range[0])
251         v2 = int(classes_range[1])
252         v3 = subject_name
253         v4 = int(age_range[0])
254         v5 = int(age_range[1])
255         view.before_and_after_search("started")
256         search.find_subjects_teachers_records_by_name_classes_age(classes_range, subject_name, age_range)
257         view.before_and_after_search("ended")

```

```

257         view.before_and_after_search("ended")
258     except:
259         view.incorrect_input_message("search")
260 elif command == "search schedule records by id, name, day":
261     try:
262         id_range = view.ask_for_values_to_search("id range").split(", ")
263         teacher_name = view.ask_for_values_to_search("teacher's name")
264         day = view.ask_for_values_to_search("day")
265         v1 = int(id_range[0])
266         v2 = int(id_range[1])
267         v3 = teacher_name
268         v4 = day
269         view.before_and_after_search("started")
270         search.find_schedule_records_by_id_name_day(id_range, teacher_name, day)
271         view.before_and_after_search("ended")
272     except:
273         view.incorrect_input_message("search")
274 elif command == "search marks records by name, mark, grade":
275     try:
276         mark_range = view.ask_for_values_to_search("mark range").split(", ")
277         subject_name = view.ask_for_values_to_search("subject name")
278         grade = view.ask_for_values_to_search("grade")
279         v1 = int(mark_range[0])
280         v2 = int(mark_range[1])
281         v3 = subject_name
282         v4 = int(grade)
283         view.before_and_after_search("started")
284         search.find_marks_records_by_name_mark_grade(mark_range, grade, subject_name)
285         view.before_and_after_search("ended")
286     except:
287         view.incorrect_input_message("search")
288 else:
289     print("Unknown command, try again!")

```

## Model.py:

471 lines (413 sloc) | 17.1 KB

```
1  class Student:
2      def __init__(self, connection):
3          self.connection = connection
4
5
6      def add_student(self, line):
7          try:
8              line_adding = line.split(", ")
9              students = [(line_adding[0], line_adding[1], line_adding[2])]
10             student_records = ", ".join(["%s"]*len(students))
11             insert_query = (
12                 f"INSERT INTO students (name, age, grade) VALUES {student_records}"
13             )
14             self.connection.autocommit = True
15             cursor = self.connection.cursor()
16             cursor.execute(insert_query, students)
17         except:
18             print("Error: student was not added!")
19
20
21     def update_student(self, line):
22         try:
23             line_editing = line.split(", ")
24             update_student = f"""
25             UPDATE
26             students
27             SET
28                 name = '{line_editing[1]}',
29                 age = '{line_editing[2]}',
30                 grade = '{line_editing[3]}'
31             WHERE
32                 id = {line_editing[0]}
33             """
34             self.connection.autocommit = True
35             cursor = self.connection.cursor()
36             cursor.execute(update_student, line_editing)
37         except:
38             print("Error: student was not updated!")
39
40
41     def delete_student(self, line_deleting):
42         try:
```

```

42         try:
43             delete_student = f"DELETE FROM students WHERE id = '{line_deleting}'"
44             self.connection.autocommit = True
45             cursor = self.connection.cursor()
46             cursor.execute(delete_student, line_deleting)
47         except:
48             print(f"Error: student with id = {line_deleting} does not exist!")
49
50
51     def generate_random_students(self, line_adding):
52         try:
53             insert_query = (
54                 f"""INSERT INTO students(name, age, grade)
55                 SELECT
56                 chr(trunc(65+RANDOM()*25)::INT)||chr(trunc(65+RANDOM()*25)::INT) AS name,
57                 trunc(RANDOM() * 15 + 6)::INT AS age,
58                 trunc(RANDOM() * 10 + 2)::INT AS grade
59                 FROM GENERATE_SERIES(1, {line_adding}) seq;"""
60             )
61             self.connection.autocommit = True
62             cursor = self.connection.cursor()
63             cursor.execute(insert_query, line_adding)
64         except:
65             print("Error: student was not generated!")
66
67
68
69     class Teacher:
70         def __init__(self, connection):
71             self.connection = connection
72
73
74         def add_teacher(self, line):
75             try:
76                 line_adding = line.split(", ")
77                 teachers = [(line_adding[0], line_adding[1], line_adding[2])]
78                 teacher_records = ", ".join(["%s"] * len(teachers))
79                 insert_query = (
80                     f"INSERT INTO teachers (name, age, work_experience) VALUES {teacher_records}"
81                 )
82                 self.connection.autocommit = True
83                 cursor = self.connection.cursor()
84                 cursor.execute(insert_query, teachers)
85             except:
86                 print("Error: teacher was not added!")

```

```

86         print("Error: teacher was not added!")
87
88     def update_teacher(self, line):
89         try:
90             line_editing = line.split(", ")
91             update_teacher = f"""
92             UPDATE
93             teachers
94             SET
95                 name = '{line_editing[1]}',
96                 age = '{line_editing[2]}',
97                 work_experience = '{line_editing[3]}'
98             WHERE
99                 id = {line_editing[0]}
100             """
101             self.connection.autocommit = True
102             cursor = self.connection.cursor()
103             cursor.execute(update_teacher, line_editing)
104         except:
105             print("Error: teacher was not updated!")
106
107     def delete_teacher(self, line_deleting):
108         try:
109             delete_teacher = f"DELETE FROM teachers WHERE id = '{line_deleting}'"
110             self.connection.autocommit = True
111             cursor = self.connection.cursor()
112             cursor.execute(delete_teacher, line_deleting)
113         except:
114             print(f"Error: teacher with id = {line_deleting} does not exist!")
115
116     def generate_random_teachers(self, line_adding):
117         try:
118             insert_query = (
119                 f"""INSERT INTO teachers(name, age, work_experience)
120                 SELECT
121                     chr(trunc(65+RANDOM()*25)::INT)||chr(trunc(65+RANDOM()*25)::INT) AS name,
122                     trunc(RANDOM() * 10 + 6)::INT AS age,
123                     trunc(RANDOM() * 10 + 10)::INT AS work_experience
124                 FROM GENERATE_SERIES(1, {line_adding}) seq;"""
125             )
126             self.connection.autocommit = True
127             cursor = self.connection.cursor()
128             cursor.execute(insert_query, line_adding)
129         except:
130             print("Error: teacher was not generated!")

```

```

130         print("Error: teacher was not generated!")
131
132
133
134
135 class Subject:
136     def __init__(self, connection):
137         self.connection = connection
138
139
140     def add_subject(self, line):
141         try:
142             line_adding = line.split(", ")
143             subjects = [(line_adding[0], line_adding[1])]
144             subject_records = ", ".join(["%s"] * len(subjects))
145             insert_query = (
146                 f"INSERT INTO teachers (name, classes_per_semester) VALUES {subject_records}"
147             )
148             self.connection.autocommit = True
149             cursor = self.connection.cursor()
150             cursor.execute(insert_query, subjects)
151         except:
152             print("Error: subject was not added!")
153
154     def update_subject(self, line):
155         try:
156             line_editing = line.split(", ")
157             update_subject = f"""
158             UPDATE
159             subjects
160             SET
161             name = '{line_editing[1]}',
162             classes_per_semester = '{line_editing[2]}'
163             WHERE
164             id = {line_editing[0]}
165             """
166             self.connection.autocommit = True
167             cursor = self.connection.cursor()
168             cursor.execute(update_subject, line_editing)
169         except:
170             print("Error: subject was not updated!")
171
172     def delete_subject(self, line_deleting):
173         try:
174             delete_subject = f"DELETE FROM subjects WHERE id = '{line_deleting}'"

```



```

174         delete_subject = f"DELETE FROM subjects WHERE id = '{line_deleting}'"
175         self.connection.autocommit = True
176         cursor = self.connection.cursor()
177         cursor.execute(delete_subject, line_deleting)
178     except:
179         print(f"Error: subject with id = {line_deleting} does not exist!")
180
181     def generate_random_subjects(self, line_adding):
182         try:
183             insert_query = (
184                 f"""INSERT INTO subjects(name, classes_per_semester)
185                 SELECT
186                 chr(trunc(65+RANDOM()*25)::INT)||chr(trunc(65+RANDOM()*25)::INT) AS name,
187                 trunc(RANDOM() * 10 + 6)::INT AS classes_per_semester
188                 FROM GENERATE_SERIES(1, {line_adding}) seq;"""
189             )
190             self.connection.autocommit = True
191             cursor = self.connection.cursor()
192             cursor.execute(insert_query, line_adding)
193         except:
194             print("Error: subject was not generated!")
195
196
197
198     class Subject_Teacher:
199         def __init__(self, connection):
200             self.connection = connection
201
202
203         def add_subjects_teachers_record(self, line):
204             try:
205                 line_adding = line.split(", ")
206                 records = [(line_adding[0], line_adding[1])]
207                 subjects_teachers_records = ", ".join(["%s" * len(records)])
208                 insert_query = (
209                     f"INSERT INTO subjects_teachers (student_id, teacher_id) VALUES {subjects_teachers_records}"
210                 )
211                 self.connection.autocommit = True
212                 cursor = self.connection.cursor()
213                 cursor.execute(insert_query, records)
214             except:
215                 print("Error: record was not added!")
216
217         def update_subjects_teachers_record(self, line):
218             try:

```

```

218         try:
219             line_editing = line.split(", ")
220             update_record = f"""
221             UPDATE
222                 subjects_teachers
223             SET
224                 subject_id = '{line_editing[1]}',
225                 teacher_id = '{line_editing[2]}'
226             WHERE
227                 id = {line_editing[0]}
228             """
229             self.connection.autocommit = True
230             cursor = self.connection.cursor()
231             cursor.execute(update_record, line_editing)
232         except:
233             print("Error: record was not updated!")
234
235     def delete_subjects_teachers_record(self, line_deleting):
236         try:
237             delete_record = f"DELETE FROM subjects_teachers WHERE id = '{line_deleting}'"
238             self.connection.autocommit = True
239             cursor = self.connection.cursor()
240             cursor.execute(delete_record, line_deleting)
241         except:
242             print(f"Error: record with id = {line_deleting} does not exist!")
243
244     def generate_random_subjects_teachers_records(self, line_adding):
245         try:
246             insert_query = (
247                 f"""INSERT INTO subjects_teachers(subject_id, teacher_id)
248                 SELECT
249                     trunc(RANDOM() * 10 + 6)::INT AS subject_id,
250                     trunc(RANDOM() * 10 + 3)::INT AS teacher_id
251                 FROM GENERATE_SERIES(1, {line_adding}) seq;"""
252             )
253             self.connection.autocommit = True
254             cursor = self.connection.cursor()
255             cursor.execute(insert_query, line_adding)
256         except:
257             print("Error: record was not generated!")
258
259
260
261
262     class Schedule:

```

```

262 class Schedule:
263     def __init__(self, connection):
264         self.connection = connection
265
266
267     def add_schedule_record(self, line):
268         try:
269             line_adding = line.split(", ")
270             record = [(line_adding[0], line_adding[1], line_adding[2], line_adding[3])]
271             schedule_records = ", ".join(["%s" * len(record)])
272             insert_query = (
273                 f"INSERT INTO scedule (day, time, subject_teacher_id, student_id) VALUES {schedule_records}"
274             )
275             self.connection.autocommit = True
276             cursor = self.connection.cursor()
277             cursor.execute(insert_query, record)
278         except:
279             print("Error: record was not added!")
280
281     def update_schedule_record(self, line):
282         try:
283             line_editing = line.split(", ")
284             update_record = f"""
285             UPDATE
286                 scedule
287             SET
288                 day = '{line_editing[1]}',
289                 time = '{line_editing[2]}',
290                 subject_teacher_id = '{line_editing[3]}',
291                 student_id = '{line_editing[4]}'
292             WHERE
293                 id = {line_editing[0]}
294             """
295             self.connection.autocommit = True
296             cursor = self.connection.cursor()
297             cursor.execute(update_record, line_editing)
298         except:
299             print("Error: record was not updated!")
300
301     def delete_schedule_record(self, line_deleting):
302         try:
303             delete_record = f"DELETE FROM scedule WHERE id = '{line_deleting}'"
304             self.connection.autocommit = True
305             cursor = self.connection.cursor()
306             cursor.execute(delete_record, line_deleting)

```

```

306         cursor.execute(delete_record, line_deleting)
307     except:
308         print(f"Error: record with id = {line_deleting} does not exist!")
309
310     def generate_random_schedule_records(self, line_adding):
311         try:
312             insert_query = (
313                 f"""INSERT INTO scedule(day, time, subject_teacher_id, student_id)
314                 SELECT
315                 CASE trunc(RANDOM() * 10)::INT
316                     WHEN 0 THEN 'monday'
317                     WHEN 1 THEN 'tuesday'
318                     WHEN 2 THEN 'wednesday'
319                     WHEN 3 THEN 'thursday'
320                     WHEN 4 THEN 'friday'
321                     ELSE 'saturday'
322                 END AS day,
323                 trunc(RANDOM() * 10 + 5)::INT||':'||trunc(RANDOM() * 10 + 5)::INT AS time,
324                 trunc(RANDOM() * 2 + 1)::INT AS subject_teacher_id,
325                 trunc(RANDOM() * 2 + 1)::INT AS student_id
326                 FROM GENERATE_SERIES(1, {line_adding}) seq;"""
327             )
328             self.connection.autocommit = True
329             cursor = self.connection.cursor()
330             cursor.execute(insert_query, line_adding)
331         except:
332             print("Error: record was not generated!")
333
334
335
336
337     class Mark:
338         def __init__(self, connection):
339             self.connection = connection
340
341         def add_mark(self, line):
342             try:
343                 line_adding = line.split(", ")
344                 marks = [(line_adding[0], line_adding[1], line_adding[2])]
345                 mark_records = ", ".join(["%s" * len(marks)])
346                 insert_query = (
347                     f"INSERT INTO marks (student_id, subject_teacher_id, mark) VALUES {mark_records}"
348                 )
349                 self.connection.autocommit = True
350                 cursor = self.connection.cursor()

```

---

```

350         cursor = self.connection.cursor()
351         cursor.execute(insert_query, marks)
352     except:
353         print("Error: mark was not added!")
354
355     def update_mark(self, line):
356         try:
357             line_editing = line.split(", ")
358             update_mark = f"""
359             UPDATE
360             marks
361             SET
362                 name = '{line_editing[1]}',
363                 subject_teacher_id = '{line_editing[2]}',
364                 mark = '{line_editing[3]}'
365             WHERE
366                 id = {line_editing[0]}
367             """
368             self.connection.autocommit = True
369             cursor = self.connection.cursor()
370             cursor.execute(update_mark, line_editing)
371         except:
372             print("Error: mark was not updated!")
373
374     def delete_mark(self, line_deleting):
375         try:
376             delete_mark = f"DELETE FROM marks WHERE id = '{line_deleting}'"
377             self.connection.autocommit = True
378             cursor = self.connection.cursor()
379             cursor.execute(delete_mark, line_deleting)
380         except:
381             print(f"Error: mark with id = {line_deleting} does not exist!")
382
383     def generate_random_marks(self, line_adding):
384         try:
385             insert_query = (
386                 f"""INSERT INTO marks(student_id, subject_teacher_id, mark)
387                 SELECT
388                     trunc(RANDOM() * 2 + 1)::INT AS student_id,
389                     trunc(RANDOM() * 2 + 1)::INT AS subject_teacher_id,
390                     trunc(RANDOM() * 10 + 5)::INT AS mark
391                 FROM GENERATE_SERIES(1, {line_adding}) seq;"""
392             )
393             self.connection.autocommit = True
394             cursor = self.connection.cursor()

```

```

394         cursor = self.connection.cursor()
395         cursor.execute(insert_query, line_adding)
396     except:
397         print("Error: mark was not generated!")
398
399
400
401
402 class Search:
403     def __init__(self, connection):
404         self.connection = connection
405
406
407     def find_subjects_teachers_records_by_name_classes_age(self, classes_range, subject_name, age_range):
408         try:
409             find_by_value_query = (
410                 f"""SELECT DISTINCT sub.name, sub.classes_per_semester, teach.name, teach.age
411                     FROM subjects sub, teachers teach, subjects_teachers st
412                     WHERE
413                         st.subject_id = sub.id
414                         AND sub.name = '{subject_name}'
415                         AND sub.classes_per_semester > {classes_range[0]}
416                         AND sub.classes_per_semester < {classes_range[1]}
417                         AND teach.age > {age_range[0]}
418                         AND teach.age < {age_range[1]}"""
419             )
420             self.connection.autocommit = True
421             cursor = self.connection.cursor()
422             cursor.execute(find_by_value_query)
423             for line in cursor.fetchall():
424                 print(line)
425         except:
426             print("Error: records were not found!")
427
428     def find_schedule_records_by_id_name_day(self, id_range, teacher_name, day):
429         try:
430             find_by_value_query = (
431                 f"""SELECT DISTINCT sc.id, sc.day, sc.time, sub.name, teach.name, stud.name
432                     FROM scedule sc, subjects sub, teachers teach, students stud, subjects_teachers st
433                     WHERE
434                         sc.id > {id_range[0]}
435                         AND sc.id < {id_range[1]}
436                         AND st.id = sc.subject_teacher_id
437                         AND sub.id = st.subject_id
438                         AND teach.id = st.teacher_id

```

```

438         AND teach.id = st.teacher_id
439         AND stud.id = sc.student_id
440         AND teach.name LIKE '%{teacher_name}%'
441         AND sc.day = '{day}'"""
442     )
443     self.connection.autocommit = True
444     cursor = self.connection.cursor()
445     cursor.execute(find_by_value_query)
446     for line in cursor.fetchall():
447         print(line)
448 except:
449     print("Error: records were not found!")
450
451 def find_marks_records_by_name_mark_grade(self, mark_range, grade, subject_name):
452     try:
453         find_by_value_query = (
454             f"""SELECT DISTINCT ma.mark, stud.name, stud.grade, sub.name, teach.name
455                 FROM subjects sub, teachers teach, subjects_teachers st, marks ma, students stud
456                 WHERE
457                     ma.mark >= {mark_range[0]}
458                     AND ma.mark <= {mark_range[1]}
459                     AND stud.id = ma.student_id
460                     AND st.id = ma.subject_teacher_id
461                     AND teach.id = st.teacher_id
462                     AND stud.grade = '{grade}'
463                     AND sub.name LIKE '%{subject_name}%'"""
464             )
465         self.connection.autocommit = True
466         cursor = self.connection.cursor()
467         cursor.execute(find_by_value_query)
468         for line in cursor.fetchall():
469             print(line)
470 except:
471     print("Error: records were not found!")

```

## View.py:

40 lines (28 sloc) | 1.17 KB

```
1  class View:
2
3      def ask_for_values_to_add(self, entity):
4          value = input(f"Enter new {entity}'s values: ")
5          return value
6
7      def added_message(self, entity):
8          print(f"+++ {entity} is added +++")
9
10     def ask_for_values_to_update(self, entity):
11         value = input(f"Enter updated {entity}'s values: ")
12         return value
13
14     def updated_message(self, entity):
15         print(f"*** {entity} is updated ***")
16
17     def ask_for_values_to_delete(self, entity):
18         value = input(f"Enter {entity}'s id: ")
19         return value
20
21     def deleted_message(self, entity):
22         print(f"--- {entity} is deleted ---")
23
24     def ask_for_values_to_generate(self, entity):
25         value = input(f"Enter number of random {entity}s")
26         return value
27
28     def generated_message(self, entity):
29         print(f"@@@ {entity}s are generated")
30
31     def ask_for_values_to_search(self, message):
32         value = input(f"Enter {message}: ")
33         return value
34
35     def before_and_after_search(self, message):
36         print(f"____search {message}____")
37
38     def incorrect_input_message(self, entity):
39         print(f"!!! incorrect values for {entity} were entered, try again !!!")
```